

FIG. 1

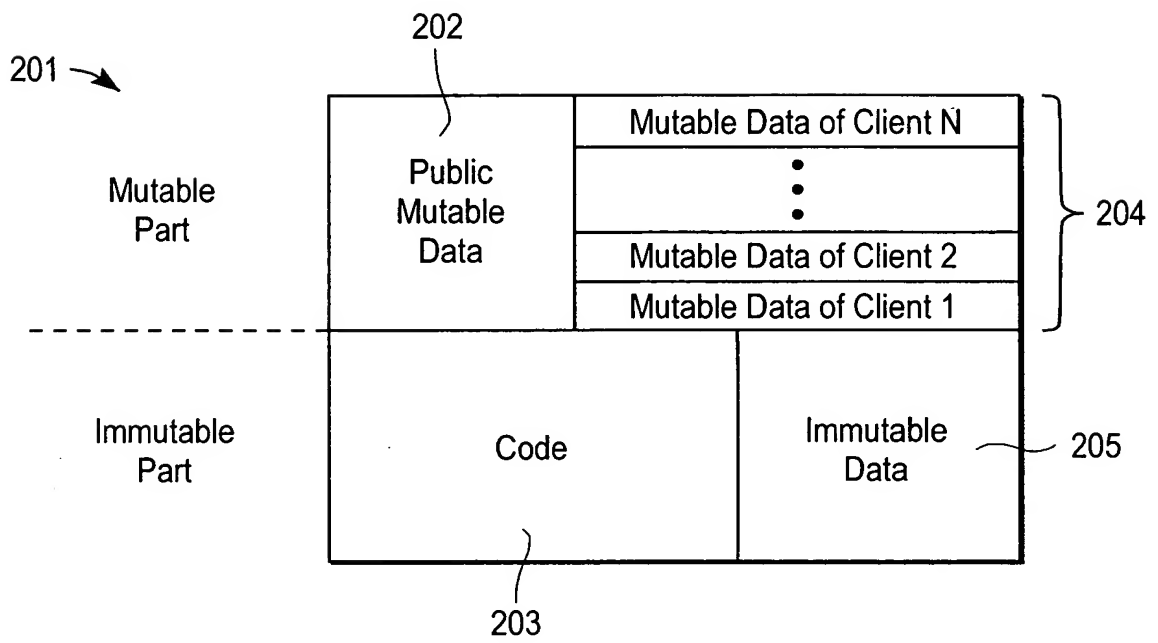


FIG. 2

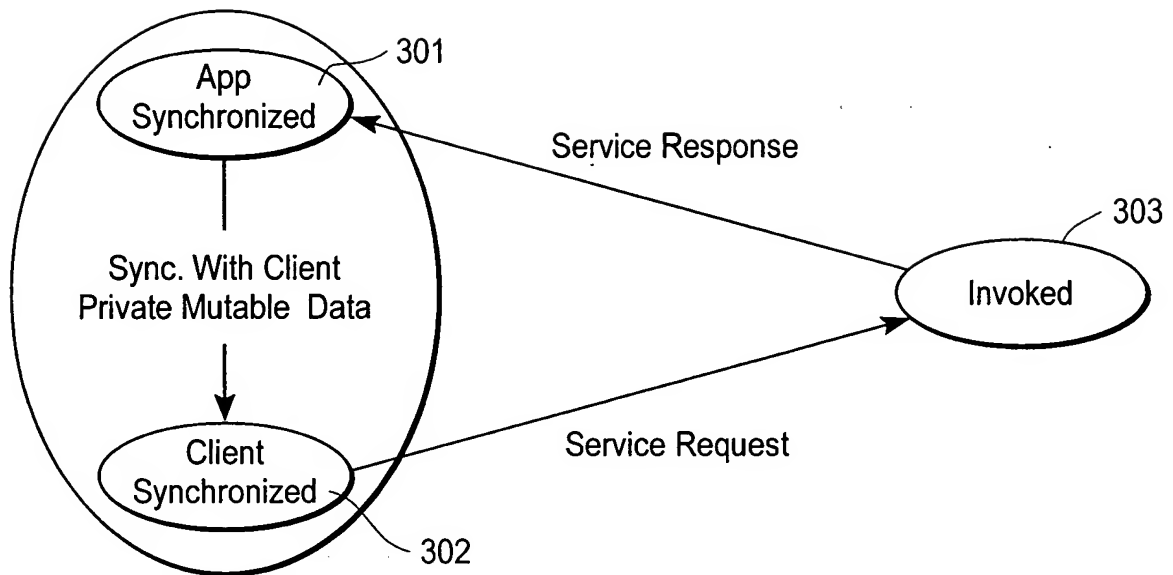


FIG. 3

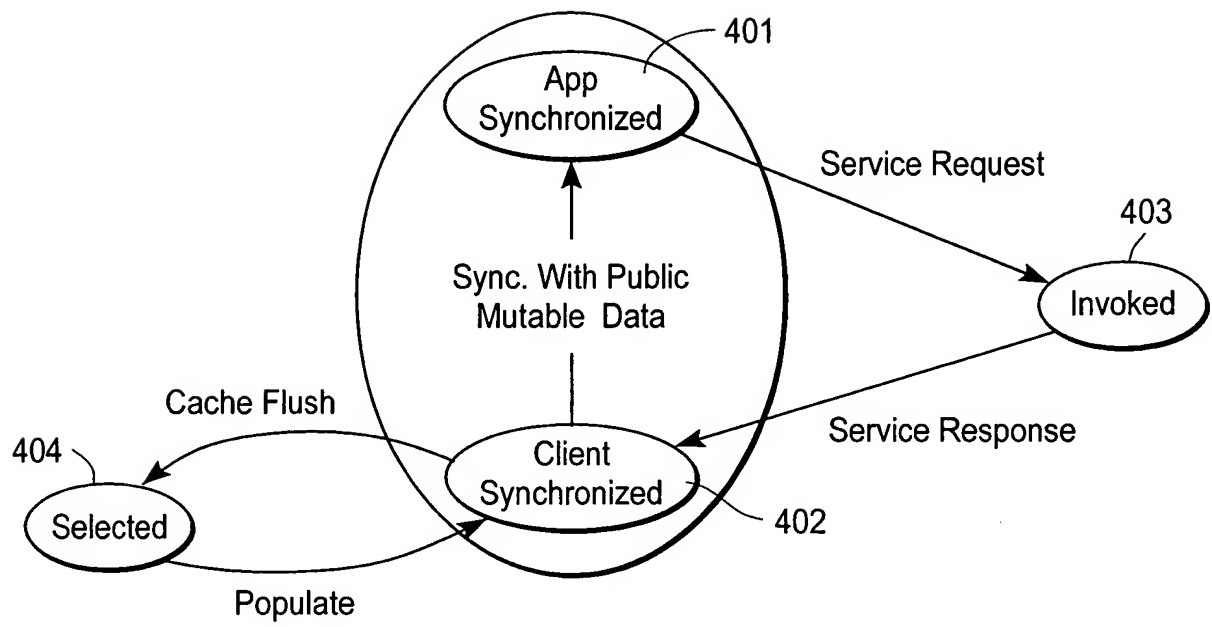


FIG. 4

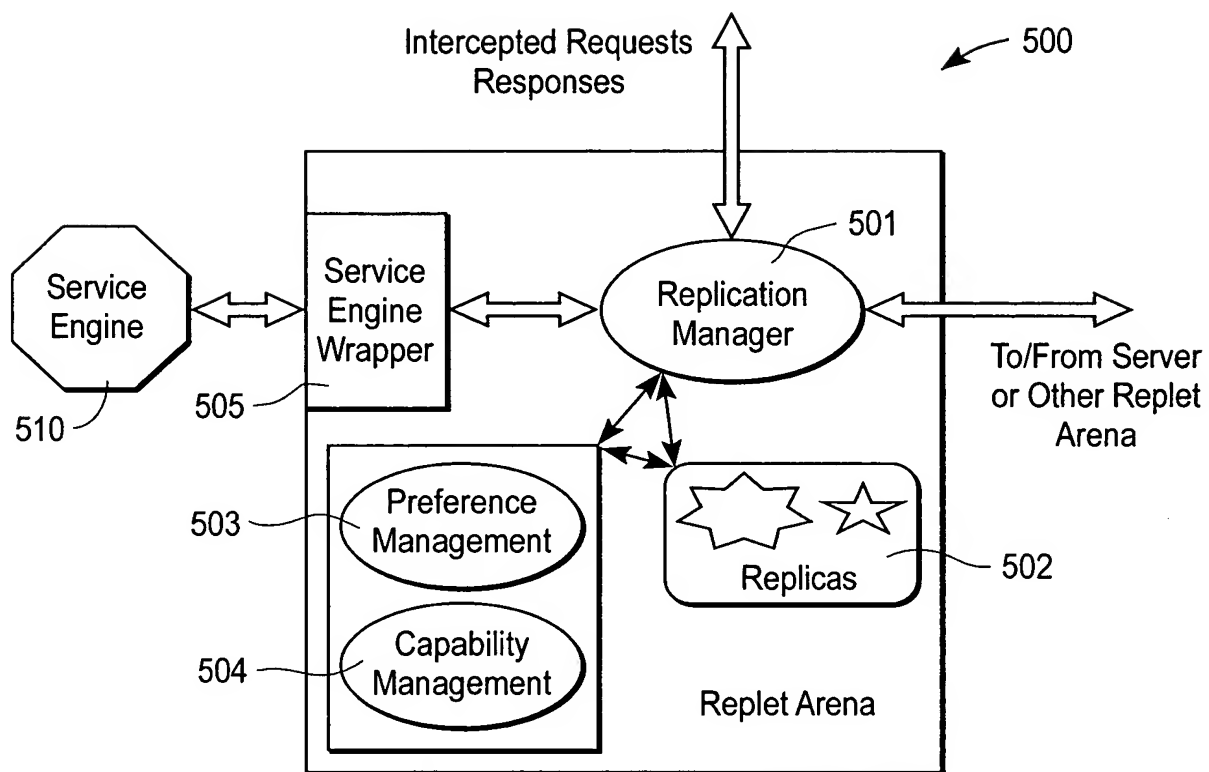


FIG. 5

```
public AdaptationSuggestion serverConsult (RepletWrapper mervlet,  
      ClientExtendedCPI cecpi, Object sessionId) {  
    boolean useClientReplica = false;  
    float confidence = 0.5f;  
    float load = RepletProfiler.getLatestCPULoad();  
  
    if (load >= RepletPreference.cpuLoadHighWaterMark) {  
        useClientReplica = true;  
        confidence = (load - RepletPreference.cpuLoadHighWaterMark) / load;  
    } else {  
        confidence = (RepletPreference.cpuLoadHighWaterMark - load) /  
            RepletPreference.cpuLoadHighWaterMark;  
        return new AdaptationSuggestion (useClientReplica, confidence);  
    }  
}
```

FIG. 6

```

public class MyAdaptationConsultant implements AdaptationConsultant {
    Preference pref = mervlet.getPolicy ();
    Profiler profiler = mervlet.getProfiler ();
    boolean useClientReplica = false;
    float confidence = 0;

    waitTimeThreshold = pref.getFloat ("WAIT_TIME_THRESHOLD");
    thrashingFactor = pref.getFloat ("THRASHING_FACTOR");
    varianceThreshold = pref.getFloat ("VARIANCE_THRESHOLD");
    float clientServiceCost = profiler.estClientServiceCost ();
    float serverServiceCost = profiler.estServerServiceCost ();
    float transmissionCost = profiler.estTransmissionCost ();

    if (mervlet.isActive()) {
        if (clientServiceCost < waitTimeThreshold) {
            useClientReplica = true;
            confidence = 1 - clientServiceCost / waitTimeThreshold;
        } else {
            float syncCost = profiler.estSyncCost (sessionID);
            float tmp = syncCost + serverServiceCost + transmissionCost;
            if (tmp * thrashingFactor < clientServiceCost)
                confidence = 1 - tmp * thrashingFactor / clientServiceCost;
            else {
                useClientReplica = true;
                confidence = 1 - clientServiceCost / (tmp * thrashingFactor);
            }
        }
    } else {
        float waitTimeVariance = profiler.estWaitTimeVariance ();
        if (serverServiceCost + transmissionCost < waitTimeThreshold
            && waitTimeVariance < varianceThreshold)
            confidence = 1 - (serverServiceCost + transmissionCost) / waitTimeThreshold;
        else {
            float activationCost = profiler.estActivationCost (sessionID);
            if (!mervlet.isInstalled ()) activationCost += profiler.estInstallationCost();
            if (activationCost + clientServiceCost < waitTimeThreshold) {
                useClientReplica = true;
                confidence = 1 - (activationCost + clientServiceCost) / waitTimeThreshold;
            } else {
                float tmp1 = transmissionCost + serverServiceCost;
                float tmp2 = (activationCost + clientServiceCost)
                    * thrashingFactor;
                if (tmp1 < tmp2) confidence = 1 - tmp1 / tmp2;
                else {
                    useClientReplica = true;
                    confidence = 1 - tmp2 / tmp1;
                }
            }
        }
    }
}

return new AdaptationSuggestion (useClientReplica, confidence);
}

```

FIG. 7

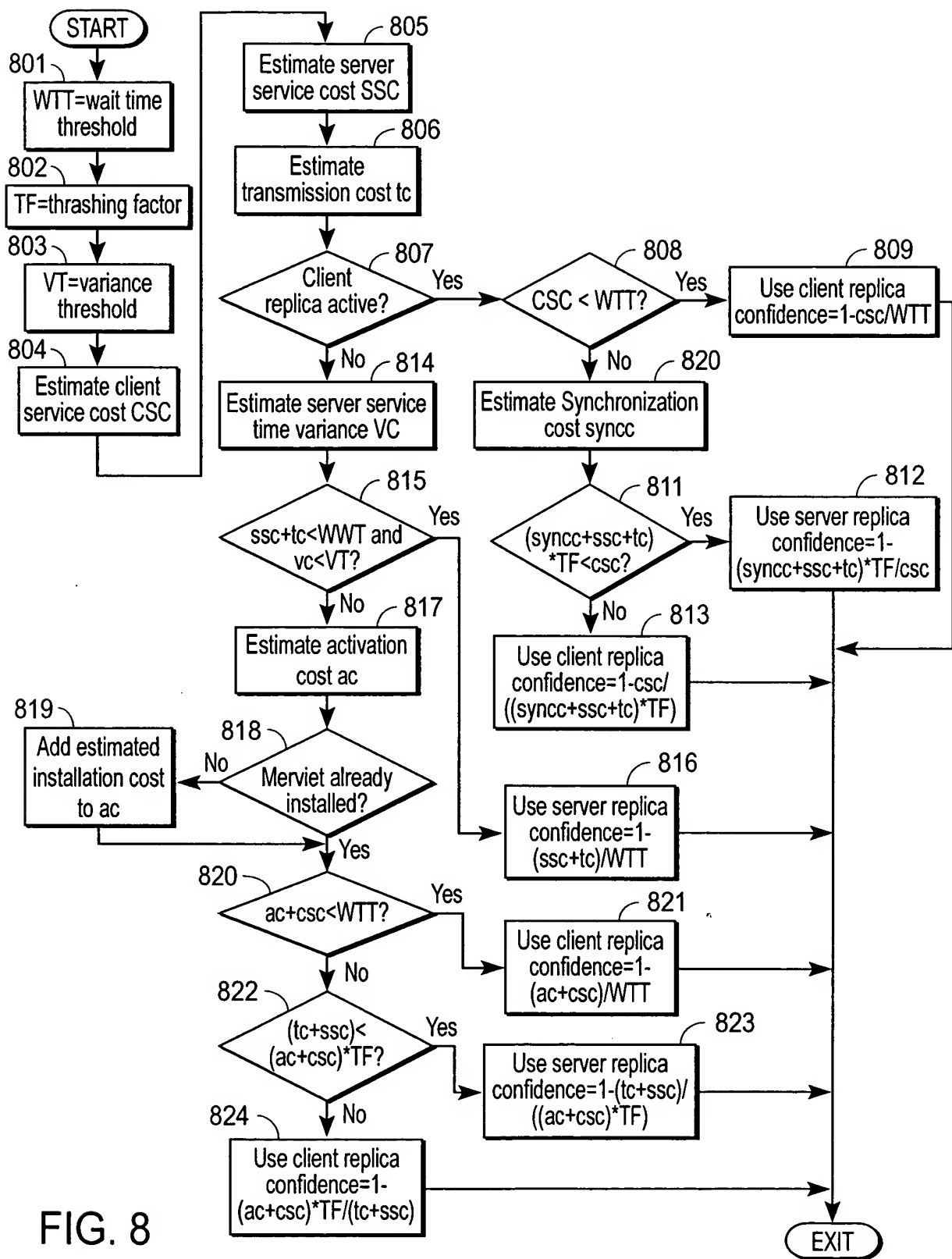


FIG. 8

```
// Get the preference derivation policy for Replet replication
DerivationPolicy dp = DerivationPolicy.get ("RepletReplication");

// find the user guideline
Guideline gline = new Guideline ("My guideline file");

// register myself as a "user", save the returned authentication code
long code = dp.register ("user", gline);

// get the derived preference
Preference pref = dp.getDerivedPreference ();

// change my wait threshold to 3 seconds, specify a priority of 5
// tell the meta policy that I'm the user, give the authentication code
pref.setGuidelineItem ("user", code, "waitThreshold", "3", 5);
```

FIG. 9

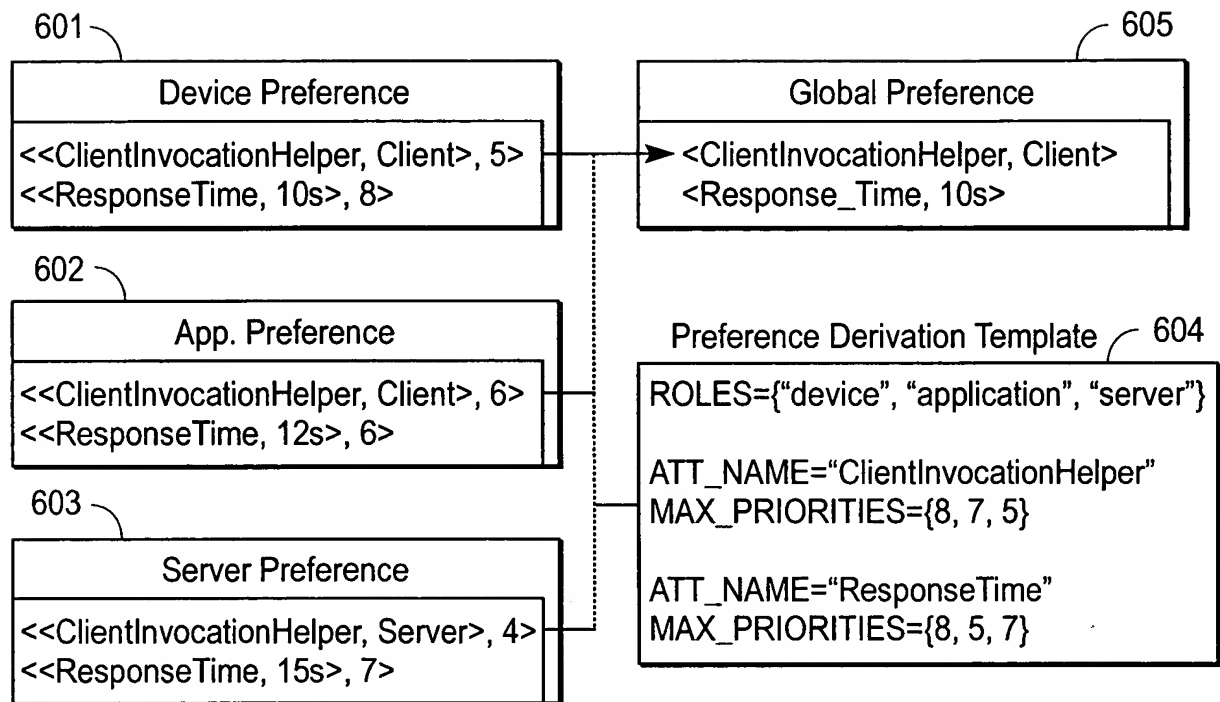


FIG. 10

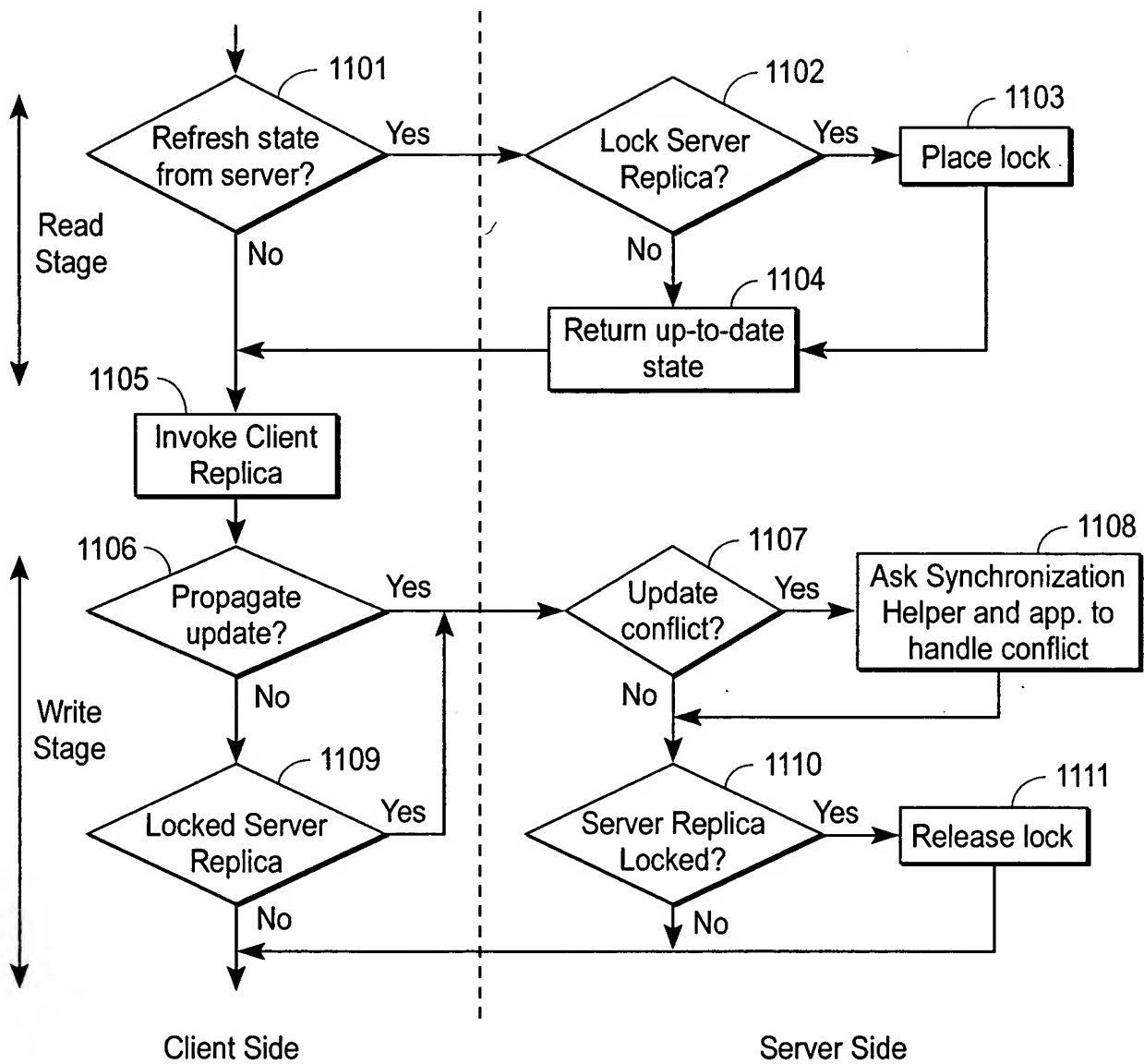


FIG. 11

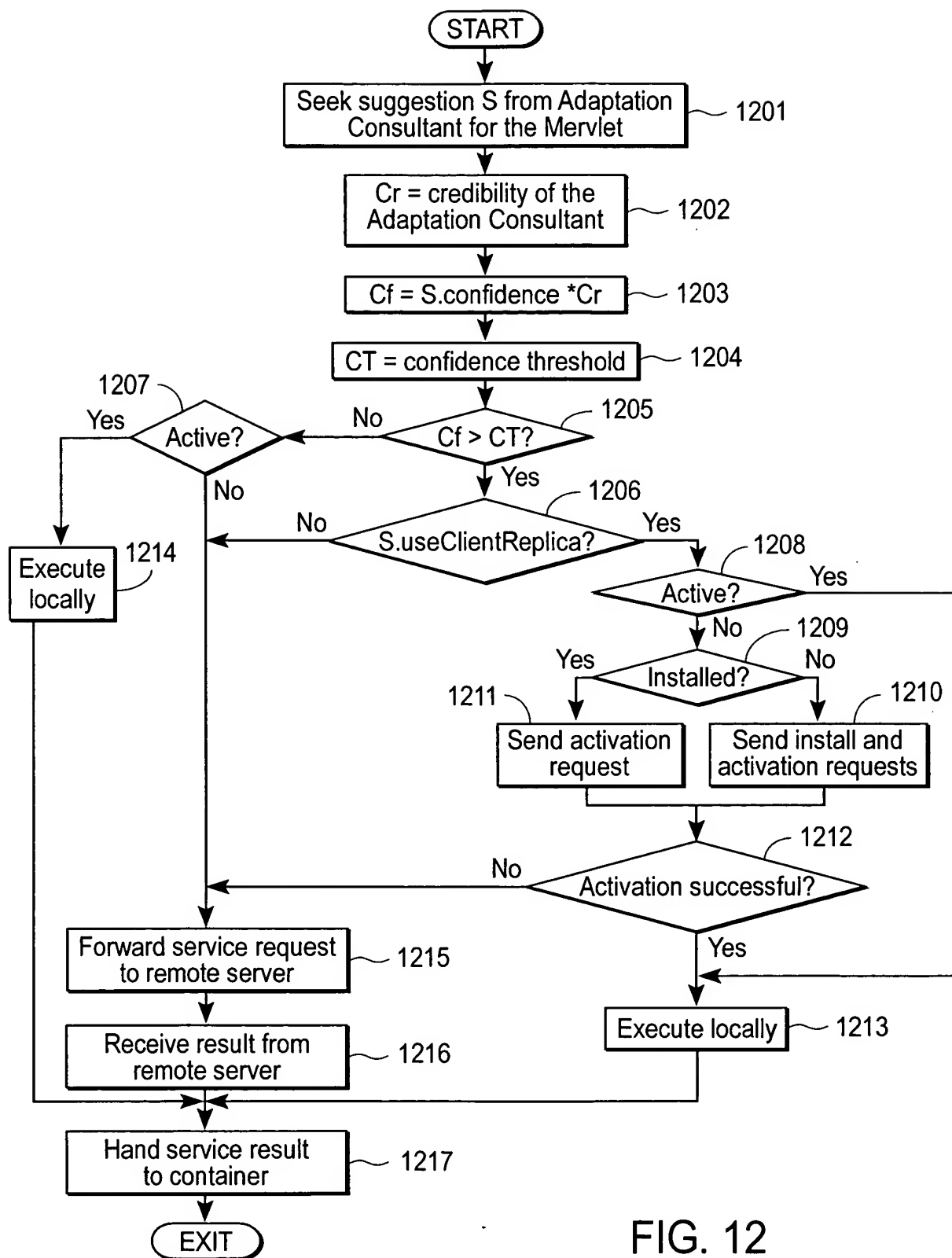
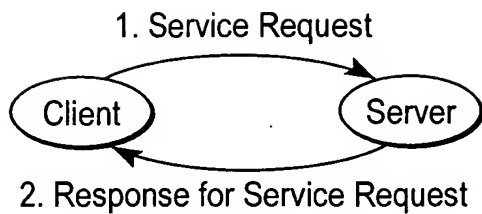
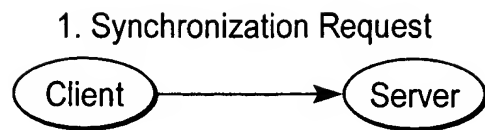


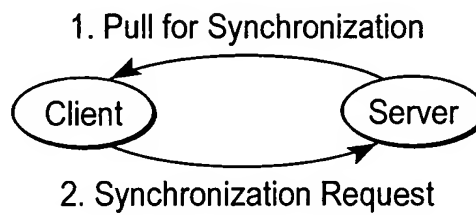
FIG. 12



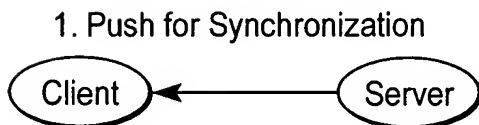
Case 1: Service Request/Response



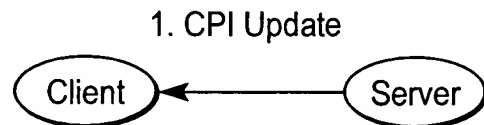
Case 2: Client actively sending sync data



Case 3: Server pulling synchronization data from client



Case 4: Server pushing synchronization data to client



Case 5: Server pushing updated CPI to client

FIG. 13

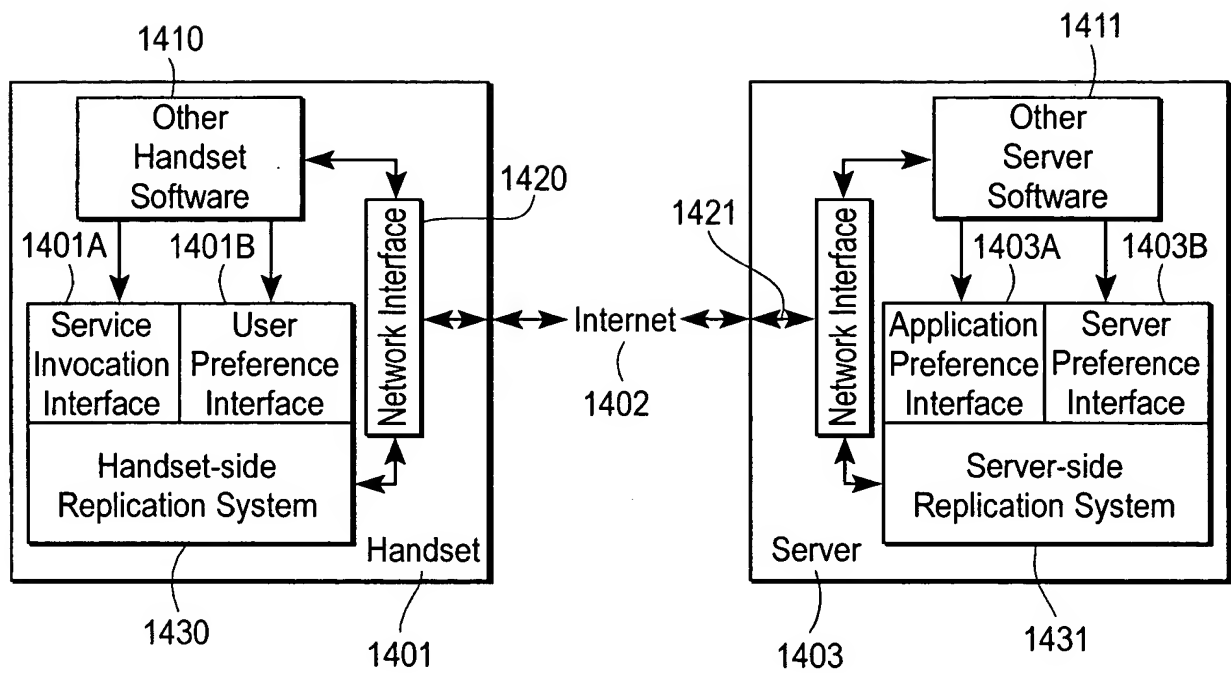


FIG. 14

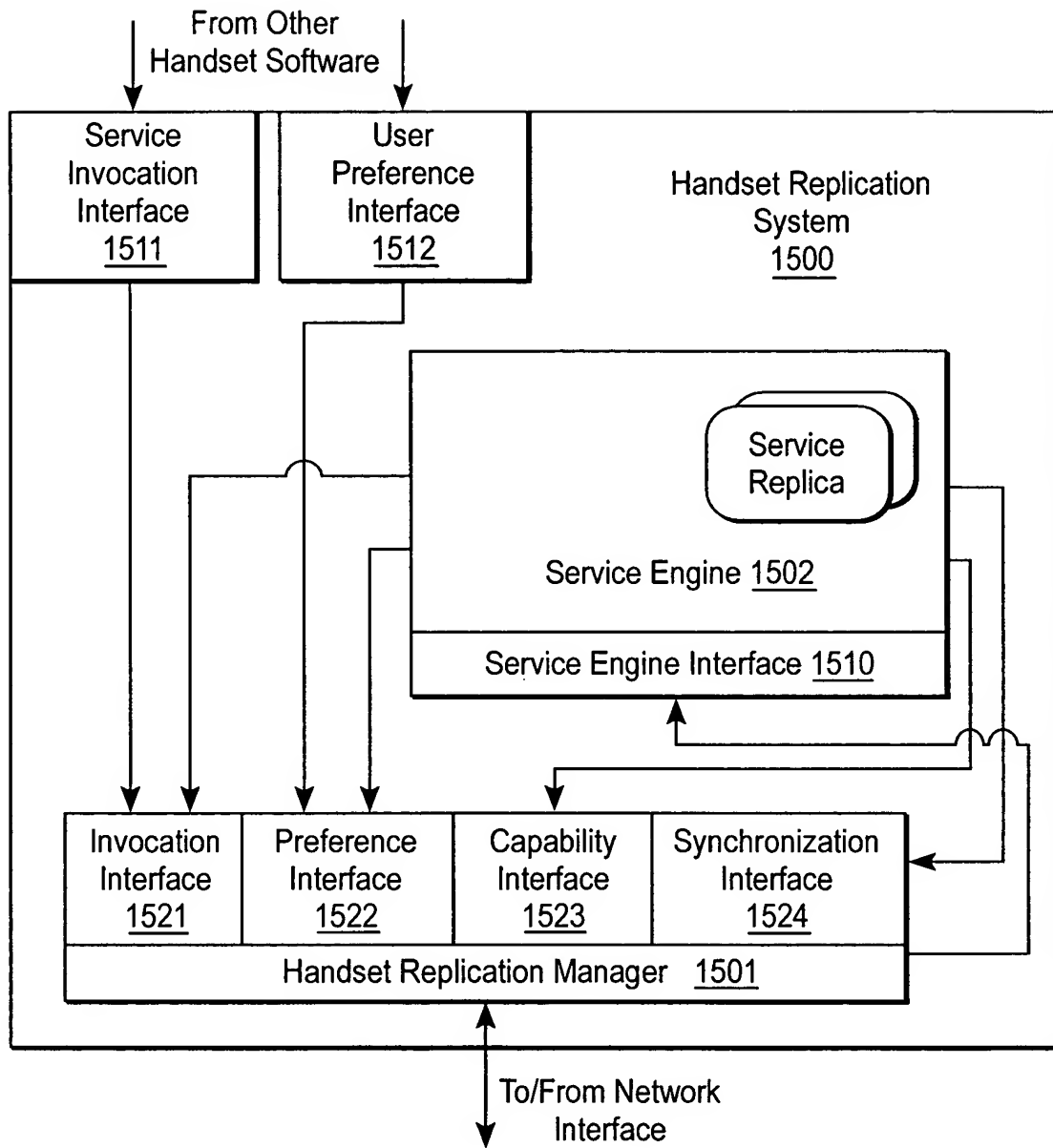


FIG. 15

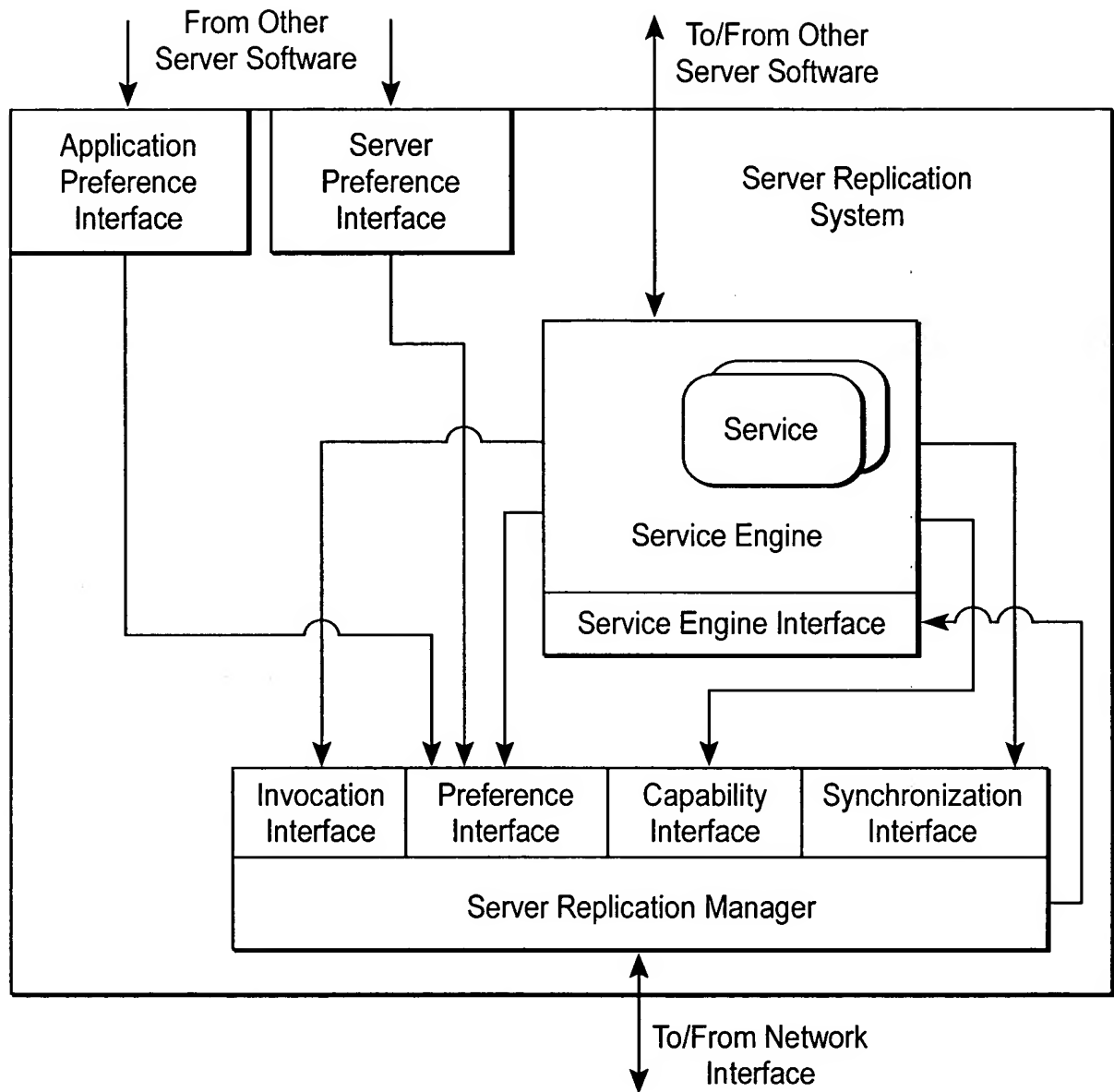


FIG. 16

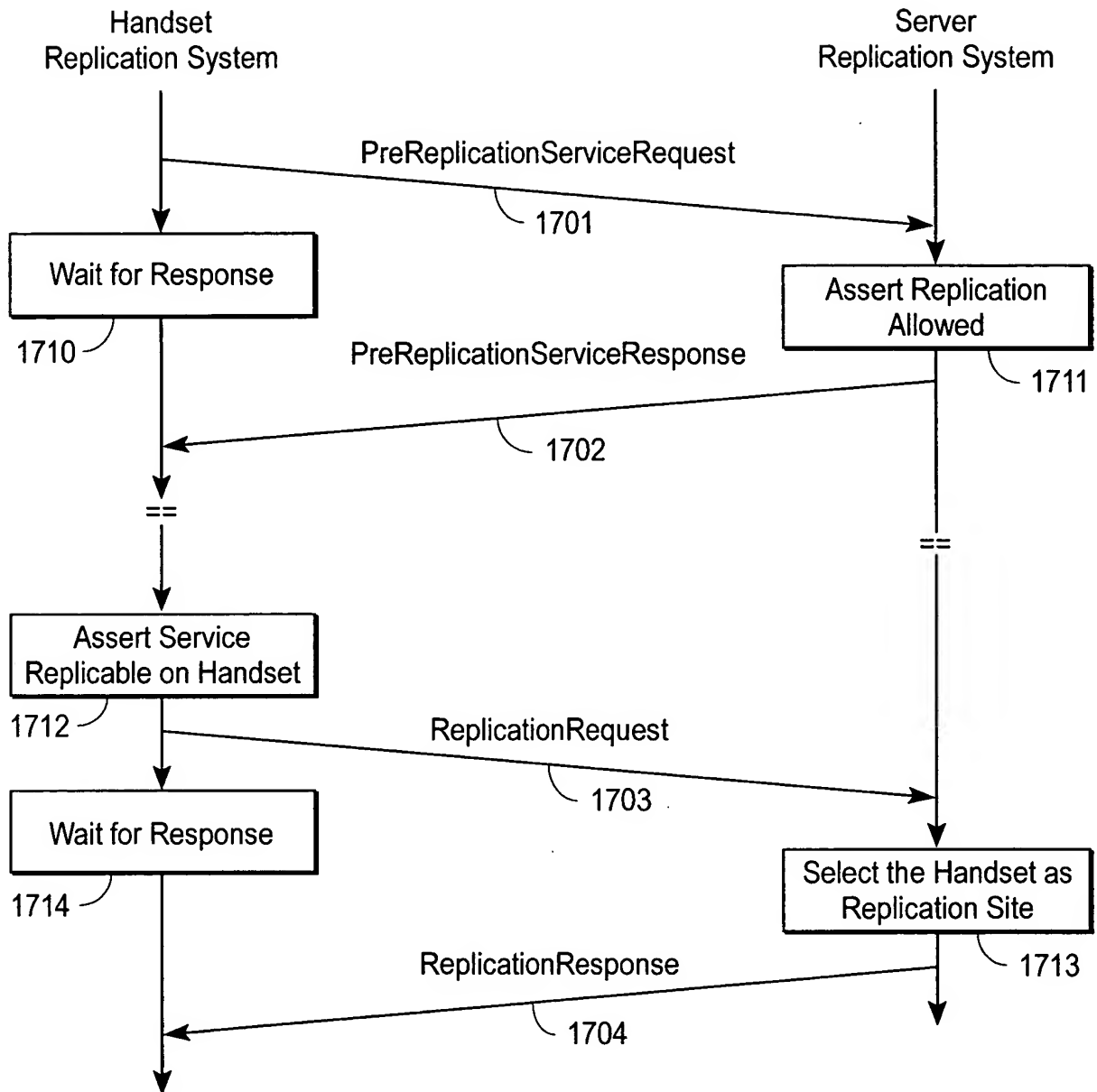


FIG. 17

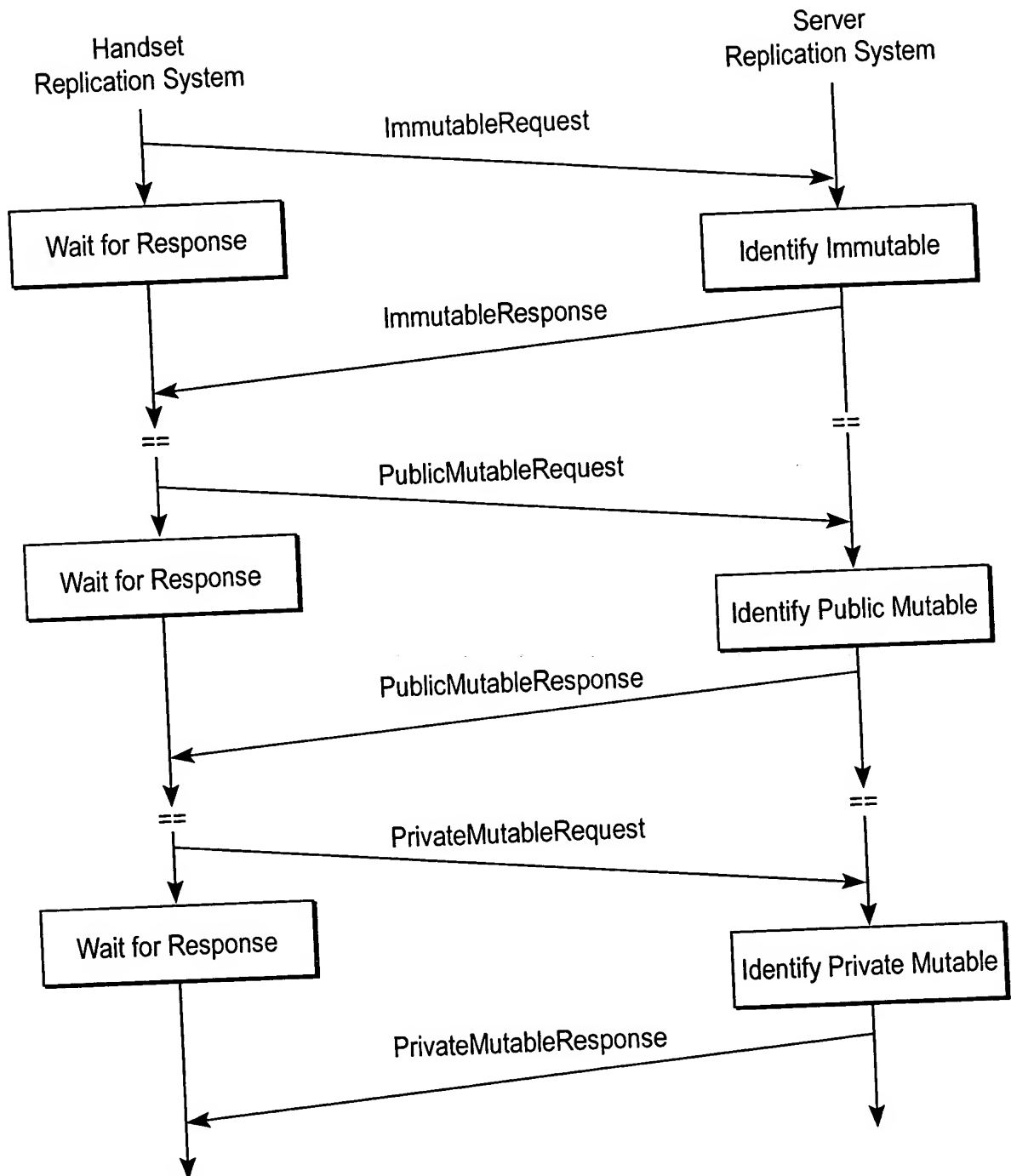


FIG. 18

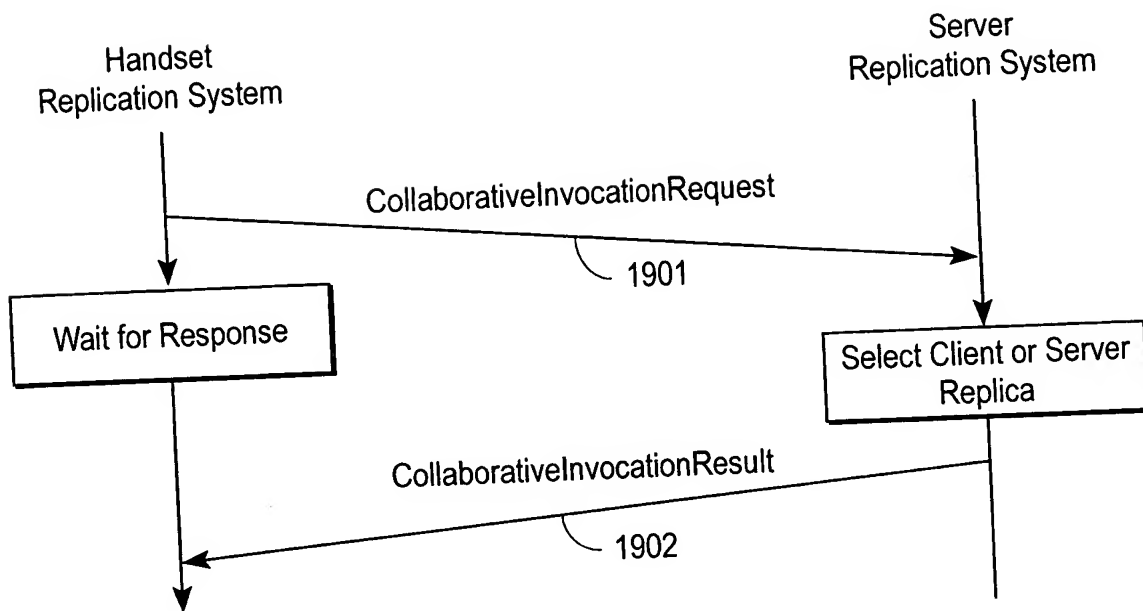


FIG. 19

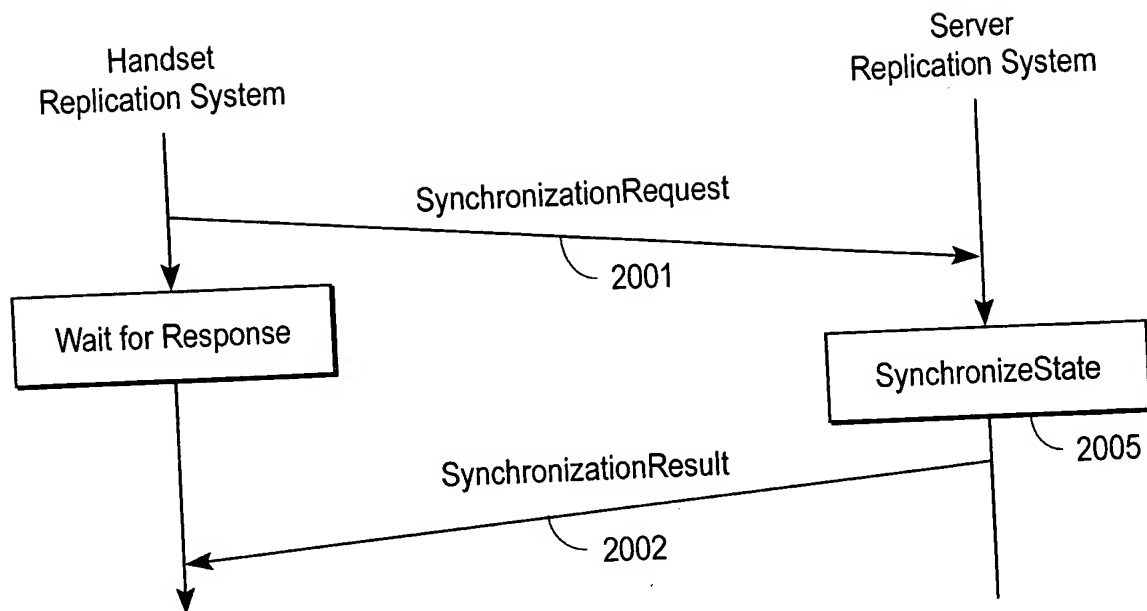


FIG. 20

2100

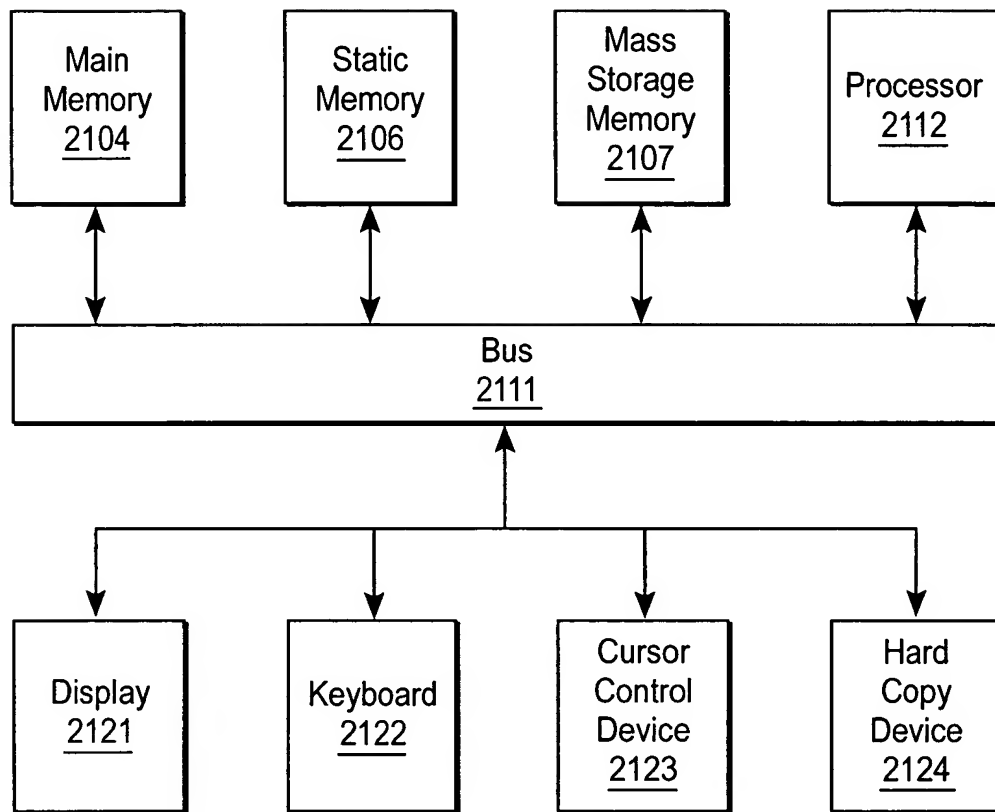


FIG. 21